

APPENDIX 2: MATLAB SOURCE CODE

HEAT FILTER - 514

```
function out = heat_flow2D( I, num_iter, time_step )
% heat_flow2D: 2D heat flow filter same as Gaussian filtering.
%
% out = heat_flow2D( I, num_iter, time_step )
% * I is the input image
% * num_iter is the number of iterations for which to run the filter
% * time_step is the size of the time step between iterations.
% * out is the output smoothed image
%
% The heat flow equation is I(t+1) = I(t) + time_step * Laplacian(I)
% Gaussian standard deviation (sigma) = sqrt( 2 * num_iter * time_step )
% or, if time_step = 0.25, num_iter = 2 * sigma^2.

[m n] = size(I);
update = zeros(m, n);
rows = 1:m;
cols = 1:n;

20 prev_rows = rows - 1;
prev_rows(1) = 1;
next_rows = rows + 1;
next_rows(m) = m;

25 prev_cols = cols - 1;
prev_cols(1) = 1;
next_cols = cols + 1;
next_cols(n) = n;

30 for iter = 1:num_iter
    update = (I(rows, next_cols) + I(next_rows,cols) - 4*I(rows,cols) + I(prev_rows, cols) +
    I(rows, prev_cols))/4;
    I = I + time_step * update;
35 end

out = I;
```

SHOCK FILTER - 518

```
function out = shock_filter_grad2D( I, num_iter, time_step, grad_thresh )
%-- 2D shock filter
%-- the equation is I(t+1) = I(t) - sign(Ixx) . | grad( I ) |
```



25315

PATENT TRADEMARK OFFICE



```

%-- sign(Ixx) is the laplace and grad(I) is the upwind derivative
% here instead of using Ixx for edges, we use magnitude of grad(I) as edge.

5 [m n] = size(I);
update = zeros(m, n);
diff = zeros(m,n);
gradmat = zeros(m, n);
gradmag = zeros(m, n);
laplace = zeros(m, n);
10 diffpos = zeros(m, n);
diffneg = zeros(m, n);
dplus = zeros( m, n);
dminus = zeros( m, n);
cond1 = zeros( m, n);
15 cond2 = zeros( m, n);

rows = 1:m;
cols = 1:n;

20 prev_rows = rows - 1;
prev_rows(1) = 1;
next_rows = rows + 1;
next_rows(m) = m;

25 prev_cols = cols - 1;
prev_cols(1) = 1;
next_cols = cols + 1;
next_cols(n) = n;

30 for iter = 1:num_iter
    %laplace = (I(rows, next_cols) + I(next_rows,cols) - 4*I(rows,cols) + I(prev_rows, cols) +
    I(rows, prev_cols))/4;
    laplace = laplace2( I );
35
    % instead of centered diff - lets use upwind
    %gradmat = 0.5*abs(I(next_rows, cols) - I(prev_rows, cols));

    %row derivative forward
40    diff = I(next_rows,cols) - I(rows,cols);
    cond1 = diff >= 0;
    cond2 = diff < 0;
    diffpos( cond1 ) = diff( cond1 );
    diffpos( cond2 ) = 0;
45    diffneg( cond2 ) = diff( cond2 );
    diffneg( cond1 ) = 0;

```



25315

PATENT TRADEMARK OFFICE



```

dplus = diffneg .* diffneg;
dminus = diffpos .* diffpos;

5   %row derivative backward
diff = I(rows,cols) - I(prev_rows,cols);
cond1 = diff >= 0;
cond2 = diff < 0;
10  diffpos( cond1 ) = diff( cond1 );
diffpos( cond2 ) = 0;
diffneg( cond2 ) = diff( cond2 );
diffneg( cond1 ) = 0;

dplus = dplus + diffpos .* diffpos;
15  dminus = dminus + diffneg .* diffneg;

%col derivative forward
diff = I(rows,next_cols) - I(rows,cols);
cond1 = diff >= 0;
20  cond2 = diff < 0;
diffpos( cond1 ) = diff( cond1 );
diffpos( cond2 ) = 0;
diffneg( cond2 ) = diff( cond2 );
diffneg( cond1 ) = 0;

25  dplus = dplus + diffneg .* diffneg;
dminus = dminus + diffpos .* diffpos;

%col derivative backward
30  diff = I(rows,cols) - I(rows,prev_cols);
cond1 = diff >= 0;
cond2 = diff < 0;
diffpos( cond1 ) = diff( cond1 );
diffpos( cond2 ) = 0;
35  diffneg( cond2 ) = diff( cond2 );
diffneg( cond1 ) = 0;

dplus = dplus + diffpos .* diffpos;
dminus = dminus + diffneg .* diffneg;
40  dplus = sqrt( dplus );
dminus = sqrt( dminus );

gradmat( laplace >= 0 ) = dplus( laplace >= 0 );
45  gradmat( laplace < 0 ) = dminus( laplace < 0 );

```



25315

PATENT TRADEMARK OFFICE

DXUC-1-1020AP

- 83 -

BLACK LOWE & GRAHAM PLLC

816 Second Avenue
Seattle, Washington 98104
206.381.3300 • F: 206.381.3301

```

gradmat(gradmat < grad_thresh) = 0;

laplace( laplace > 0 ) = 1;
laplace( laplace < 0 ) = -1;
5
update = -gradmat .* laplace;
I = I + time_step .* update;
end

10 out = I;

```

INTENSITY CLUSTERING – 422

```

function min_img = cluster_image( in_img, num_clusters, sample_factor, num_iterations )
% cluster_image: Cluster an intensity image using the k-means algorithm.
15
%
% out_img = cluster_image( in_img, num_clusters, sample_factor, num_iterations )
% * in_img is the input intensity image to be clustered
% * num_clusters is the number of output clusters desired
% * sample_factor is a factor by which to downsample the input data. If 1, then no
20 sampling, if 2, then reduces
% input data by half.
% * num_iterations is the maximum number of iterations that the algorithm should be run
if it does not converge.
% usually it converges in about 2-6 iterations - so you can set this to be about 20.
25 % * min_img is the output image consisting of pixels lying on the cluster with the
minimum mean.
%
% A simple k-means algorithm is used for this clustering.

30 convergence_threshold = 0.05;

in_data = downsample( in_img(:), sample_factor );

minval = min( in_data );
35 maxval = max( in_data );

incval = ( maxval - minval )/num_clusters;

%cluster boundaries
40 boundaries = zeros(1, num_clusters + 1 );
boundaries(1) = (minval-0.5);
boundaries(num_clusters+1) = (maxval+0.5);
for i = 2:num_clusters
    boundaries(i) = boundaries(i-1) + incval;

```



25315

PATENT TRADEMARK OFFICE

DXUC-1-1020AP

- 84 -

BLACK LOWE & GRAHAM PLLC

816 Second Avenue
Seattle, Washington 98104
206.381.3300 • F: 206.381.3301

```

end

a = 1; b = [0.5 0.5];
centers = zeros( 1, num_clusters );
5 centers = filter( b, a, boundaries );
centers = centers( 2:end );
old_centers = centers;

for iter = 1:num_iterations
10 %lets quantize the data according to the boundaries
for j = 1:num_clusters
    cdata = in_data( in_data <= boundaries(j+1) & in_data > boundaries(j) );
    centers(j) = mean(cdata);
15 end

%lets recompute the boundaries
newb = filter( b, a, centers );
newb( 1 ) = (minval-0.5);
20 boundaries = [newb, (maxval+0.5)];

%centers
diff = norm( ( centers - old_centers ) ./ old_centers );
if ( diff <= convergence_threshold )
25 %disp(sprintf('converged in %d iterations', iter));
    break;
end

old_centers = centers;
30 end

min_thresh1 = boundaries( 1 );
min_thresh2 = boundaries( 2 );
min_img = ( in_img <= min_thresh2 ) & ( in_img > min_thresh1 );
35 min_img = reshape( min_img, size( in_img ) );

```

SPATIAL GRADIENTS – 526

```

function [xgrad,ygrad,grad_mag] = spatial_gradients( in_img )
40 % spatial_gradients: Compute spatial gradients of the 2D input image
%
% [xgrad,ygrad,grad_mag] = spatial_gradients( in_img )
% * in_img is the input input image
% * xgrad is the x-direction gradient

```



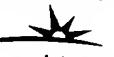
25315

PATENT TRADEMARK OFFICE

DXUC-1-1020AP

- 85 -

BLACK LOWE & GRAHAM PLLC


816 Second Avenue
Seattle, Washington 98104
206.381.3300 • F: 206.381.3301

```

% * ygrad is the y-direction gradient
% * grad_mag is the gradient magnitude
%
5  kernx = [-1 1];
kerny = [-1; 1];
xgrad = imfilter( in_img, kernx );
ygrad = imfilter( in_img, kerny );
10 grad_mag = sqrt( xgrad.^2 + ygrad.^2 );

```

HYSTERESIS THRESHOLD – 530

```

function out = hysteresis_threshold( grad_mag, PercentNonEdge, FractionLow )
% hysteresis_threshold: Perform hysteresis thresholding by automatically computing two
15 thresholds
%
% out = hysteresis_threshold( grad_mag, PercentNonEdge, FractionLow )
% * grad_mag is gradient magnitude image to be thresholded
% * PercentNonEdge is the percentage of non-edge pixels desired (0.97 works good)
20 % * FractionLow is the ratio between the lower and the upper threshold (0.5 works good)

%lets try to figure out the thresholds
maxgrad = max( grad_mag(:) );
25 [c x] = hist( grad_mag(:), ceil( maxgrad ) );
[m n] = size( grad_mag );
high_thresh = min( find( cumsum( c ) > PercentNonEdge * m * n ) );

%lets threshold the gradients
30 out = hysteresis_threshold_core( grad_mag, FractionLow * high_thresh, high_thresh );

function out = hysteresis_threshold_core( img, t1, t2 )
%do a hysteresis thresholding of an image at a lower threshold of t1 and an upper threshold
35 of t2
%basically, we first do a thresholding at the lower value
%then we do a connected component
%then we pick those components which have at least one value above t2
%here, we are going to do it a little differently,
40 % 1. we will first threshold at t2
% 2. we will select one point each from t2
% 3. then we will threshold at t1
% 4. for each point from t2, we will use bwlabel/bwselect to find the component on the t1
image
45 %

```



25315

PATENT TRADEMARK OFFICE

```

%lets first threshold at higher threshold t2
t2_thresh = img > t2;

5 [r c] = find( t2_thresh );

%lets now threshold at lower thresh t1
t1_thresh = img > t1;

10 out = bwselect( t1_thresh, c, r, 8 );

```

MATCHING EDGES FILTER - 538

```

function [out_filled, out_edg2] = find_matching_vertical_edges( in_edg, ygrad )
% find_matching_vertical_edges: Find matching edges in the vertical direction
15 %
% out_filled = find_matching_vertical_edges( in_edg, ygrad )
% * in_edg is the input edge image
% * ygrad is the y-direction gradient
% * out_filled is the output filled region between edges
20 %

out_ygrad = ygrad;
out_ygrad( in_edg == 0 ) = 0;

25 dist_thresh = 5;

%for every line, select the closest edge pairs
[npts nline] = size( in_edg );
out_edg2 = zeros( size( in_edg ) );
30 out_filled = out_edg2;

for line = 1:nline
    fw_pts = find( out_ygrad(:,line) < 0 );
    bw_pts = find( out_ygrad(:,line) > 0 );
35    lfw = length( fw_pts );
    lbw = length( bw_pts );
    if ( lfw == 0 | lbw == 0 )
        continue;
    end
40    if ( ( lfw == 1 ) & ( lbw == 1 ) )
        if ( bw_pts(1) > fw_pts(1) )
            out_edg2( fw_pts(1), line ) = 1;
            out_edg2( bw_pts(1), line ) = 2;
            out_filled( fw_pts(1):bw_pts(1), line ) = 1;

```



25315

PATENT TRADEMARK OFFICE

DXUC-I-1020AP

- 87 -

BLACK LOWE & GRAHAM PLLC

816 Second Avenue
Seattle, Washington 98104
206.381.3300 • F: 206.381.3301

```

    end
    continue;
end

5    votes_fw = zeros( size( fw_pts ) );
    votes_bw = zeros( size( bw_pts ) );

%for every front wall point find the closest backwall point
for fw = 1:lfw
10    curr_fw = fw_pts( fw );
    min_bw = 0;
    min_dist = 1000000;
    for bw = 1:lbw
        curr_dist = bw_pts( bw ) - curr_fw;
15    if ( curr_dist > dist_thresh & curr_dist < min_dist )
        min_bw = bw;
        min_dist = curr_dist;
    end
    end
20    if ( min_bw > 0 )
        votes_fw( fw ) = votes_fw( fw ) + 1;
        votes_bw( min_bw ) = votes_bw( min_bw ) + 1;
    end
end
25

%for every bacak wall point find the closest frontwall point
for bw = 1:lbw
    curr_bw = bw_pts( bw );
    min_fw = 0;
30    min_dist = 1000000;
    for fw = 1:lfw
        curr_dist = curr_bw - fw_pts( fw );
        if ( curr_dist > dist_thresh & curr_dist < min_dist )
            min_fw = fw;
            min_dist = curr_dist;
        end
    end
    if ( min_fw > 0 )
        votes_fw( min_fw ) = votes_fw( min_fw ) + 1;
        votes_bw( bw ) = votes_bw( bw ) + 1;
40    end
end

%at this point, the fw and bw with votes greater than 1 should be matching
45    good_fw_pts = fw_pts( votes_fw > 1 );
    good_bw_pts = bw_pts( votes_bw > 1 );

```



25315

PATENT TRADEMARK OFFICE

DXUC-1-1020AP

- 88 -

BLACK LOWE & GRAHAM PLLC



816 Second Avenue
Seattle, Washington 98104
206.381.3300 • F: 206.381.3301

```

if ( length( good_fw_pts ) ~= length( good_bw_pts ) )
    disp('match not found');
else
5    for i = 1:length( good_fw_pts )
        out_edg2( good_fw_pts(i), line ) = 1;
        out_edg2( good_bw_pts(i), line ) = 2;
        out_filled( good_fw_pts(i):good_bw_pts(i), line ) = 1;
    end
10   end
end

```

CLOSE & OPEN - 546 & 550

```

15  function out = close3D_brick( in, size1, size2, size3 )

        tmp = dilate3D_brick(in, size1, size2, size3 );
        out = erode3D_brick(in, size1, size2, size3 );

20  function out = open3D_brick( in, size1, size2, size3 )

        tmp = erode3D_brick(in, size1, size2, size3 );
        out = dilate3D_brick(in, size1, size2, size3 );

25  function out = erode3D_brick( in, size1, size2, size3 )
        % erode3D_brick: Erode a binary image with a brick shaped structuring element
        %
        % out_img = erode3D_brick( in, size1, size2, size3 )
        % * in_img is the input binary image to be dilated
        % * size1 is the size of the brick along the row dimension
        % * size2 is the size of the brick along the column dimension
        % * size3 is the size of the brick along the slice dimension
        %
        % A separable order-statistic filter with the ordinal as min is used.

35      [nr nc np] = size( in );
        out = in;

        for p = 1:np
            %lets first do the dilation along rows
            out(:,:,p) = ordfilt2( out(:,:,p), 1, ones( size1, 1 ) );

40            %lets first do the dilation along columns
            out(:,:,p) = ordfilt2( out(:,:,p), 1, ones( 1, size2 ) );
        end

```



25315

PATENT TRADEMARK OFFICE

```

%now lets do it along the third dimension
%first permute dimension
tmp = permute( out, [3, 2, 1] );
5   for r = 1:nr
      tmp(:,:,r) = ordfilt2( tmp(:,:,r), 1, ones( size3, 1 ) );
    end

10  out = permute( tmp, [3, 2, 1] );

function out = dilate3D_brick( in, size1, size2, size3 )
15  % dilate3D_brick: Dilate a binary image with a brick shaped structuring element
  %
  % out_img = dilate3D_brick( in, size1, size2, size3 )
  % * in_img is the input binary image to be dilated
  % * size1 is the size of the brick along the row dimension
  % * size2 is the size of the brick along the column dimension
  % * size3 is the size of the brick along the slice dimension
  %
20  % A separable order-statistic filter with the ordinal as max is used.

  [nr nc np] = size( in );
  out = in;

25  for p = 1:np
    %lets first do the dilation along rows
    out(:,:,p) = ordfilt2( out(:,:,p), size1, ones( size1, 1 ) );

    %lets first do the dilation along columns
30  out(:,:,p) = ordfilt2( out(:,:,p), size2, ones( 1, size2 ) );
  end

35  %now lets do it along the third dimension
  %first permute dimension
  tmp = permute( out, [3, 2, 1] );
  for r = 1:nr
    tmp(:,:,r) = ordfilt2( tmp(:,:,r), size3, ones( size3, 1 ) );
  end

40  out = permute( tmp, [3, 2, 1] );

```

FILTER DEEP REGIONS – 554

```

45  function out = filter_deep_regions( in, depthThreshold )
  % filter_deep_regions: Filter out regions that start beyond the depth threshold

```



25315

PATENT TRADEMARK OFFICE

- 90 -

BLACK LOWE & GRAHAM PLLC

816 Second Avenue
Seattle, Washington 98104
206.381.3300 • F: 206.381.3301

DXUC-1-1020AP

```

%
% out = filter_deep_regions( in, depthThreshold )
% * in is the input binary image
% * depthThreshold is the threshold beyond which regions are not acceptable
5 % * out is the output filtered binary image

[ccmp n] = bwlabel( in );
props = regionprops( ccmp, 'BoundingBox' );
out = zeros( size(in) );
10 j = 0;
for i = 1:n
    if ( props(i).BoundingBox(2) < depthThreshold )
        out = out + ( ccmp == i );
        j = j+1;
    end
15 end
end

```

HEAD DIAMETER RANGE – 730

```

20 function rad_range = radrange_lookup( age, imgParams )
%lookup the radius range in pixels given the gestational age and the
%resolution

25 mean_bpd = bpd_lookup( age );
mean_rad = 0.5 * mean_bpd / imgParams.AxialResolution;
rad_range = [mean_rad - ( 0.3 * mean_rad )
            mean_rad - ( 0.2 * mean_rad ),
            mean_rad - ( 0.1 * mean_rad ),
            mean_rad,
30            mean_rad + ( 0.1 * mean_rad ),
            mean_rad + ( 0.2 * mean_rad )
            mean_rad + ( 0.3 * mean_rad )];

35 function bpd_out = bpd_lookup( age )
%lookup the bpd in mm given the gestational age

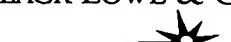
40     %BPD Table wks; BPD in mm
     bpd_table = [14, 26;
                  15, 30;
                  16, 33.5;
                  17, 36.5;
                  18, 40;
                  19, 43;
                  20, 46;
                  21, 49.5;
45                  22, 52.5;

```



25315

PATENT TRADEMARK OFFICE



```

23, 55.5;
24, 58;
25, 61;
26, 64;
5 27, 66.5;
28, 69;
29, 72;
30, 74;
31, 76.5;
10 32, 79;
33, 81;
34, 83.5;
35, 86;
36, 88;
15 37, 90;
38, 92;
39, 94;
40, 96;
20 ];
20
bpd_out = 0;
if ( age < 14 )
    return;
end
25
bpd_out = 100;
if ( age > 40 )
    return;
end
30
index = round( age - 14 + 1 );
bpd_out = bpd_table(index,2);

```

HEAD EDGE DETECTION – 734

```

35 function edg2 = find_head_regions( in_edg, ygrad, wall_edg, region_img, raddrange )
%use ygrads to find head regions

IntenThresh = 15;
DistThresh = 12;
40
ygrad( in_edg == 0 ) = 0;

45 [npts nline] = size( ygrad );
edg2 = wall_edg;

```



25315

PATENT TRADEMARK OFFICE



```

%lets go over each line and find potential head locations
%potential head locations will have
% a. good front wall and back wall pairs identified by the wall_edg input
5 % b. a matching positive gradient before the front wall - within a short distance
% c. a matching negative gradient after the back wall - within a short distance
for line = 1:nline

10 neg_grad_pts = find( ygrad(:,line) < 0 );
pos_grad_pts = find( ygrad(:,line) > 0 );
ln = length( neg_grad_pts );
lp = length( pos_grad_pts );

15 fw = find( wall_edg(:,line) == 1 );
bw = find( wall_edg(:,line) == 2 );
lfw = length(fw);
lbw = length(bw);

20 if ( lfw > 0 & lbw > 0 & lfw == lbw )
    for i = 1:lfw
        dist = bw(i) - fw(i);
        %if distance is beyond the rad range, set to zero
        if ( dist < 2*radrange(1) | dist > 2*radrange(end) )
            edg2(fw(i),line) = 0;
25        edg2(bw(i),line) = 0;
            continue;
        end

30        %if mean intensity is not at least 80% dark
        if ( mean( region_img(fw(i):bw(i), line ) ) < 0.8 )
            edg2(fw(i),line) = 0;
            edg2(bw(i),line) = 0;
            continue;
        end

35        %if within DistThresh of front wall is not a positive gradient point - else expand
        diffs = fw(i) - pos_grad_pts;
        if ( sum( diffs > 0 & diffs < DistThresh ) == 0 )
            edg2(fw(i),line) = 0;
40        else
            if ( fw(i) > 3 & fw(i) < npts - 3 )
                edg2( (fw(i)-3):(fw(i)+3), line ) = 1;
            end
        end

45        %if within DistThresh of back wall is not a negative gradient point

```



25315

PATENT TRADEMARK OFFICE

DXUC-1-1020AP

- 93 -

BLACK LOWE & GRAHAM PLLC

816 Second Avenue
Seattle, Washington 98104
206.381.3300 • F: 206.381.3301

```

1      diffss = neg_grad_pts - bw(i);
2      if ( sum( diffss > 0 & diffss < DistThresh ) == 0 )
3          edg2(bw(i),line) = 0;
4      else
5          if ( bw(i) > 3 & bw(i) < npts - 3 )
6              edg2( (bw(i)-3):(bw(i)+3), line) = 2;
7              end
8          end
9      end
10     else
11         edg2(:,line) = 0;
12     end
13 end

```

15 POLAR HOUGH TRANSFORM – 738

```

14 function out = hough_circle_polar( edg_img, radii, xducerOffset, phiAngles )
15 %compute a hough transform for circle finding on a pre-scan converted image
16 %the function goes through every edge pixel on the image and adds an accumulator
17 %corresponding to every radii
18
19 nradii = length( radii );
20 [nrow ncol] = size( edg_img );
21
22 out = zeros( nrow, ncol, nradii );
23
24 angles = 0:18:360;
25 angles = pi*angles/180;
26 cosa = cos( angles );
27 sina = sin( angles );
28
29 prows = cosa;
30 pcolls = cosa;
31
32 len = length( angles );
33 hlen = len / 2;
34
35 %go over every edge pixel
36 for row = 1:nrow
37     for col = 1:ncol
38         curr_edg = edg_img( row, col );
39         if ( curr_edg > 0 )
40             for rad = 1:nradii
41                 [prows, pcolls] = draw_polar_circle( row, col, radii(rad), cosa, sina, xducerOffset,
42 phiAngles, 0 );
43                 %xcoor = round(col + cosrad(rad,:));
44             end
45         end
46     end
47 end

```



25315

PATENT TRADEMARK OFFICE

DXUC-I-1020AP

- 94 -

BLACK LOWE & GRAHAM PLLC

816 Second Avenue
Seattle, Washington 98104
206.381.3300 • F: 206.381.3301

```

%ycoor = round(row + sinrad(rad,:));

for i = 1:len
  if ( prows(i) > 0 & prows(i) <= nrow & pcols(i) > 0 & pcols(i) <= ncol )
5    curr_out = out( prows(i), pcols(i), rad );
    %out( prows(i), pcols(i), rad ) = curr_out + 1;

    %if it is the front wall, then the center should be below it - on the lower half
    of the circle
10   if ( curr_edg == 1 & prows(i) > row )
        out( prows(i), pcols(i), rad ) = curr_out + 1;
        end

    %if it is the back wall, then the center should be above it - on the upper half of
15   the circle
        if ( curr_edg == 2 & prows(i) < row )
            out( prows(i), pcols(i), rad ) = curr_out + 1;
            end
        end %if
20   end % for i
        end % for rad
    end % if curr_edg
    end %for col
end %for row
25

function [prows, pcols] = draw_polar_circle( pcent_row, pcent_col, rad, costheta, sintheta,
xducerOffset, phiAngles, draw )
  %draw a circle in polar coordinates
30  % given a center row and column in polar coordinates, a radius in
  % cartesian, and cosine and sines of theta angles
  % yoff is the transducer offset
  % phiAngles is the list of phiAngles

35  if ( draw )
    x1 = rad * costheta + ( pcent_row + xducerOffset ) * sin( phiAngles( pcent_col ) );
    y1 = rad * sintheta + ( pcent_row + xducerOffset ) * cos( phiAngles( pcent_col ) );
  else
    x1 = -rad * costheta + ( pcent_row + xducerOffset ) * sin( phiAngles( pcent_col ) );
40  y1 = -rad * sintheta + ( pcent_row + xducerOffset ) * cos( phiAngles( pcent_col ) );
  end

  %now convert to polar
45  rads = sqrt( x1.^2 + y1.^2 );
  phis = atan2( x1, y1 );

```



25315

PATENT TRADEMARK OFFICE

DXUC-1-1020AP

- 95 -

BLACK LOWE & GRAHAM PLLC

816 Second Avenue
Seattle, Washington 98104
206.381.3300 • F: 206.381.3301

```
prows = round(rads - xducerOffset );
pcols = interp1( phiAngles, 1:length(phiAngles), phis, 'nearest' );
```

5

FILL CIRCLE REGION - 746

```
function out_image = fill_polar_circle( pcent_row, pcent_col, rad, xducerOffset, phiAngles,
10 nrow, ncol )
% fill the inside of a circle in polar coordinates
% given a center row and column in polar coordinates, a radius in
% cartesian, and cosine and sines of theta angles
% yoff is the transducer offset
% phiAngles is the list of phiAngles
15
out_image = zeros( nrow, ncol );
sinc = sin( phiAngles( pcent_col ) );
cosc = cos( phiAngles( pcent_col ) );
rad2 = rad^2;
20 lhs = 0;
for x = 1:nrow
    for y = 1:(ncol-1)
        lhs = x^2 + pcent_row^2 - 2 * x * pcent_row * ( sin(phiAngles(y))*sinc +
25 cos(phiAngles(y))*cosc );
        if ( lhs <= rad2 )
            out_image( x, y ) = 1;
        end
    end
30 end
```

35



25315

PATENT TRADEMARK OFFICE

- 96 -

DXUC-1-1020AP

BLACK LOWE & GRAHAM PLC


816 Second Avenue
Seattle, Washington 98104
206.381.3300 • F: 206.381.3301